

# Algorithms for alpha-rate domination problems on weighted graphs

Danica Vukadinović Greetham <sup>\*1</sup>, Anush Poghosyan<sup>†1</sup>, and Nathaniel Charlton<sup>‡1,2</sup>

<sup>1</sup>Centre for the Mathematics of Human Behaviour, Department of Mathematics and Statistics, University of Reading, UK,

<sup>2</sup>CountingLab Ltd, Reading, UK

November 26, 2015

## Abstract

In this article, we investigate a domination set problem variant on vertex-weighted graphs. In the last few years, several algorithms have been presented for solving the minimum alpha and alpha-rate domination problem (also known as the positive influence dominating sets problem) on simple graphs. We recently proposed an algorithm for alpha-rate domination on weighted graphs based on randomised rounding of the solution of a linear programming formulation of this problem. Due to the use of linear programming, such an algorithm could be relatively time consuming for larger graphs. Here, we propose a new version using the divide and conquer technique, which uses a graph's community structure to create a solution from the solutions obtained on denser subgraphs (with some adjustments, if necessary). We also investigate greedy techniques for this problem using three different initial vertex selection strategies. We compare two different randomised rounding and three greedy algorithms on three different families of randomly generated graphs and on four real-world graphs obtained from a Twitter mentions network. Our results show that on dense random graphs the divide and conquer technique produces results comparable in total weight to the unembellished randomised rounding method, but significantly faster. For graphs with intrinsic modular structure, the divide and conquer technique actually produces better results. When the running time is prioritised over the optimality of results, two of the three explored greedy algorithms strategies perform better than the simple strategy of picking always vertices with the smallest weights. Also, greedy techniques outperform randomised algorithms on the very sparse Twitter graphs, and on the random dense graphs for high thresholds.

**Keywords**— weighted graphs, alpha-rate domination, positive influence dominating sets, randomised rounding, planted  $l$ -partition graphs

## 1 Introduction

Studies of interplay between networks' structure and dynamical processes on them are becoming more and more popular due to the fact that a lot of complex systems that we interact with every

---

<sup>\*</sup>d.v.greetham@reading.ac.uk

<sup>†</sup>a.poghosyan@reading.ac.uk

<sup>‡</sup>n.a.charlton@reading.ac.uk

day can be modelled by networks or graphs. Examples include infrastructure such as power, water or road networks, human communication networks, social media, network based behaviour change interventions [22] and so on. In some cases, we are interested to identify a set of vertices or edges from which we can communicate to all the vertices in a given graph, or from which we can control or dominate that graph. A set of vertices/individuals such that all other or indeed all individuals are connected to that set is called a *dominating set* in graph theory. The related optimisation problem of finding a dominating set of minimal size is NP-complete [13]. From the 1950s onward, different variants of this problem have been investigated. Based on problems in ad-hoc communications networks,  $k$ -domination was explored where each vertex not in the dominating set needs to have at least  $k$  neighbours in the dominating set. Similarly, two versions having a parameter for the percentage of neighbours that need to be in a dominating set were proposed:  $\alpha$  domination (where each vertex needs to have at least  $\alpha * 100$  percent of its neighbours in the dominating set) and  $\alpha$ -rate domination [12] where each vertex, including ones in the dominating set, needs to have at least  $\alpha * 100$  percent of its neighbours in the dominating set. Again, finding minimum cardinalities of  $\alpha$  and  $\alpha$ -rate dominating sets is NP-complete. In this work, we investigate different algorithms for the  $\alpha$ -rate dominating set problem on weighted graphs, thus looking into a more general problem. Weighted graphs usually allow more realistic interpretations, and models and algorithms for weighted cases have wider application potential.

Our motivation for the weighted version of the  $\alpha$ -rate domination problem comes mainly from the health and well-being related behaviour change context. Often, for an intervention to work for an individual, it is important to have a support network (see e.g. [14]), so that positive messages can come from multiple sources. For this reason, an intervention designer might want to identify a support subset of the whole social network, which will be part of the intervention. It might be that the “best” candidates (from a structural perspective) for such a support subset are not feasible to be a part of intervention for various reasons: they do not have the desired attributes, or they do not have the capability or time to invest in the intervention. This can be represented by assigning a cost to be part of the intervention to each vertex. Then, the main task is to find the most cost effective subset from which we can control or support the network.

In the next section we give an overview of the relevant previous work. In Section 2, we present three greedy algorithm variants based on different strategies for vertex selection into the weighted alpha-rate dominating set. The randomised algorithm using a linear programming formulation of the problem from [23] is given and its new version that exploits the community structure of a graph is proposed in order to improve its running time in Section 3. We analyse the results obtained from the application of implemented greedy and randomised algorithms to three families of randomly generated graphs and Twitter mentions networks in Section 4.3. Finally, we discuss our results and give some pointers to future work in Section 4.4.

## 1.1 Previous work

Due to their suitability to a wide range of applications in networks design and control, variants of domination problems have been studied thoroughly. This includes a study of corresponding computational complexities for different variants and development of exact and approximation algorithms. The widely explored variants include the basic dominating set problem and its weighted version where weights are on vertices. The minimum weighted dominating set problem is one of the classic NP-hard optimisation problems in graph theory. Approximation algorithms for a special type of graphs, unit disk graphs with weights on vertices, were investigated in [27]. A generalisation of

the domination set problem on vertex-weighted graphs, where the direct connections are replaced with shortest paths corresponding to some measure  $f$  defined on the vertices of a graph, was explored in [6]. The authors have used randomised rounding to prove the approximation ratio of  $O(\log \Delta')$  for their algorithm, where  $\Delta'$  is the maximum cardinality of the sets of vertices that can be dominated by any single vertex through the defined shortest paths. In [5] the maximum spanning star forest problem, the complement problem of domination set, is discussed and an 0.71-approximation algorithm for this problem is given. In the vertex-weighted case, the ratio was 0.64. Molnar et al. [21] proposed probabilistic dominating set selection strategies for large heterogeneous non-weighted graphs and explored how the structure of graphs influences performances of degree dependent probabilistic method based approximation algorithms and greedy algorithms.

One generalisation of the domination problem, the so called  $k$ -dominating set problem, requires each vertex not in the dominating set to have at least  $k$  neighbours in the set. Another variant, the so called  $k$ -tuple domination problem requires each vertex in the graph (even those in the dominating set) to have at least  $k$  neighbours in the dominating set. These problems were relevant especially for ad-hoc and wireless networks routing (having more than one neighbour in a dominating set was providing more reliable connection). In [10] the author shows that a greedy approach for the minimum  $k$ -dominating set problem leads to an approximation ratio of  $\ln(\Delta + k) + 1 < \ln(\Delta) + 1.7$ , where  $\Delta$  is the maximum degree of the graph. Klasing and Laforest [19] proved the hardness of the  $k$ -tuple domination problem, even in restricted families of graphs and presented several interesting approximation algorithms.

Another generalisation, the  $\alpha$ -domination problem, was introduced by Dunbar et al. in [9], where each vertex not in the dominating set is required to have at least  $\alpha * 100$  percents of neighbours in the dominating set. Similarly, the concept of  $\alpha$ -rate domination [12] requires each vertex in the graph to have at least  $\alpha * 100$  percents of neighbours in the dominating set. Both the  $\alpha$  and  $\alpha$ -rate domination problems are proven to be NP-complete. New upper bounds and randomised algorithms for finding the  $\alpha$  and  $\alpha$ -rate domination sets in terms of the parameter  $\alpha$  and graph vertex degrees on undirected simple finite graphs are provided by using the probabilistic method in [11] and [12].

Wang et al. [24] investigated the propagation of influence in the context of social networks. They introduced new variants of domination such as the positive influence dominating set (PIDS) and total positive influence dominating set (TPIDS). Actually, the definitions of PIDS and TPIDS problems are equivalent to  $\alpha$ -dominating and  $\alpha$ -rate dominating set problems respectively for the special case when  $\alpha = 1/2$ . Dinh et al. [8] have generalised PIDS and TPIDS by allowing any  $0 < \alpha < 1$ , thus considering  $\alpha$ -dominating and  $\alpha$ -rate dominating set problems and presenting a linear time exact algorithm for trees, and approximation algorithms for PIDS and TPIDS within a factor  $\ln \Delta + O(1)$ , where  $\Delta$  is the maximum degree of the graph.

A smaller number of studies in domination parameters consider weighted graphs in particular. A variant of the weighted dominating set problem — the weighted minimum independent  $k$ -domination (WMkD) problem — was studied by Yen in [26]; an algorithm linear in the number of vertices of the input graph for the WMkD problem on trees was given.

In our previous work [23], we discussed alpha-rate domination on vertex-weighted graphs. An algorithm based on randomised rounding of a linear programming formulation of the problem is given, and we have proven that its approximation ratio is  $O(\log_2(\Delta))$ , where  $\Delta$  is the maximum degree of the graph.

## 2 Greedy algorithms

In this section, we consider greedy techniques for approximately solving the minimum weighted alpha-rate dominating set problem. We denote with  $\bar{d}_v$  and  $w_v$  the closed degree (degree plus one, including a vertex itself) and the weight of a vertex  $v \in V$  respectively. All the vertices that  $v$  is connected to together with  $v$  are called the closed neighbourhood of  $v$  and denoted with  $N[v]$ .

The Algorithm 1 below describes a generic greedy algorithm to find a low-weighted alpha-rate dominating set  $D$ .

**Input:** A graph  $G$ , a real number  $\alpha$ ,  $0 < \alpha \leq 1$   
**Output:** An  $\alpha$ -rate (total positive influence) dominating set  $D$  of  $G$

- 1: Initialize  $D = \emptyset$ ; { Form a set  $D \subseteq V(G)$ }
- 2: **while** there exists  $v \in V(G)$  s.t.  $r = |N[v] \cap D| < \alpha \bar{d}_v$  **do**
- 3:   set  $C = N[v] - D$ ;
- 4:   Initialize  $W = \emptyset$ ;
- 5:   **for all** vertices  $v \in C$  **do**
- 6:     compute  $w_v$  the sum of weights of its closed neighbourhood;
- 7:      $W = W \cup \{w_v\}$ ;
- 8:   **end for**
- 9:   Sort  $W$  using strategy S[1,2,3]
- 10:   Add the first  $\lceil \alpha \bar{d}_v \rceil - r$  vertices from  $W$  into  $D$
- 11: **end while**
- 12: **return**  $D$ ; {  $D$  is a low-weight  $\alpha$ -rate dominating set }

Algorithm 1: Greedy algorithm for finding a low-weight  $\alpha$ -rate dominating set

As is expected with a greedy process, this does not necessarily yield the optimal solution. We consider and implement three different strategies on initial selection of vertices to be added to the  $\alpha$ -rate dominating set. Those are:

**S1:** sorting vertices in vertex weight ascending order;

**S2:** sorting by using a combination of degree and sum of weights of closed neighbourhoods, thus sorting in  $\frac{w_v}{\bar{d}_v}$  ascending order;

**S3:** sorting in  $\frac{w_v}{w_{N[v]}}$  ascending order, where  $w_v$  is the weight, and  $w_{N[v]}$  is the sum of weights of the neighbourhood for vertex  $v$ .

The first strategy S1 focuses on optimising weight only. The second strategy S2 tries to balance minimising weight with minimising the size of the dominating set. Finally, the third strategy S3 is based on reasoning that it should be beneficial to take ‘light’ vertices with ‘heavy’ neighbourhoods as then less heavy neighbours will be needed in the dominating set.

In all cases we need to keep track of  $r = |N[v] \cap D|$  for each  $v \in V(G)$  only up to  $r = \lceil \alpha \bar{d}_v \rceil$ . Since we may need to browse through all the neighbours of vertices in  $V$ , in total it can take  $O(n^2)$  steps to calculate all the necessary  $|N[v] \cap D|$ ’s for each vertex  $v \in V(G)$ . Then computing and sorting a sum of weights of closed neighbourhood for each vertex can take  $O(n^2 \log n)$  steps in the worst case. Hence, in total, the set  $D$  can be computed in  $O(n^2 \log n)$  steps.

### 3 Randomised algorithms

#### 3.1 Algorithm RR

Recently, we proposed an approximation algorithm for the minimum weighted  $\alpha$ -rate dominating set problem [23]. A linear programming relaxation of the original problem was used to obtain a preliminary solution. A randomised rounding of that solution was then repeated a number of times in order to obtain a feasible solution. The idea was based on the techniques used by Chen et al. [6] for simple domination with measure functions (where adjacency may be replaced with limited length paths) on weighted graphs.

Let us assume that for every vertex  $v_i$ ,  $1 \leq i \leq n$  the variable  $x_i$  has the following meaning:  $x_i = 1$  if  $v_i$  is contained in the  $\alpha$ -rate dominating set and  $x_i = 0$  otherwise. We consider the following linear programming relaxation LP of an integer program IP:

$$\begin{aligned} \min \quad & \sum_{i=1}^n w_i x_i \\ \text{s.t.} \quad & \sum_{v_j \in N[v_i]} x_j \geq \lceil \alpha \bar{d}_{v_i} \rceil, \quad \forall v_i \in V \\ & 0 \leq x_i \leq 1, \quad 1 \leq i \leq n. \end{aligned}$$

As we know LP is polynomial-time solvable and we can compute an optimal solution  $\{\hat{x}_i\}_{1 \leq i \leq n}$ . If we denote with  $IP_{OPT}$  an optimal solution of the corresponding integer program IP we have that

$$IP_{OPT} \geq \sum_{i=1}^n w_i \hat{x}_i. \quad (1)$$

We obtain a candidate IP solution  $\{x_i\}_{1 \leq i \leq n}$  by using randomised rounding, setting  $x_i = 1$  with probability  $\hat{x}_i$  and 0 otherwise. Let  $D$  be the set of vertices that are assigned ones after rounding, i.e.  $D = \{v_i | x_i = 1, 1 \leq i \leq n\}$ .

In the next step we estimate the probability that  $D$  is a feasible solution for IP. For any vertex  $v \in V$ , with  $\bar{d}_v$  neighbours, let  $k = \lceil \alpha \bar{d}_v \rceil$ . We know that  $\sum_{v_i \in N[v]} \hat{x}_i \geq k$ , and  $\forall x_i, 0 \leq \hat{x}_i \leq 1$ . Now, the probability that  $v_i$  is  $\alpha$ -rate dominated is equal to

$$Pr(v_i \text{ is } \alpha\text{-rate dom.}) = 1 - (Pr(v_i \text{ is not } \alpha\text{-rate dom.})).$$

We can look at the number of neighbours of  $v$  (including  $v$  itself) which are in  $D$  as the sum of  $\bar{d}_v$  independent trials, random processes, where the success probability of each trial  $i$  is equal to  $\hat{x}_i$ . Thus this sum,  $|N[v] \cap D|$ , follows Poisson's binomial distribution [25] with parameters  $\hat{x}_1, \dots, \hat{x}_{\bar{d}_v}$ . Let  $S = \{1, 2, \dots, \bar{d}_v\}$ , and  $\mathcal{F}_k = \{A | A \subseteq S, |A| = k\}$  denote all subsets of  $S$  with exactly  $k$  members where we are going over all possible combinations. Then  $|\mathcal{F}_k| = \binom{\bar{d}_v}{k}$ . Let  $A^C$  denote the complementary set, i.e.  $S \setminus A$ .

$$Pr(v_i \text{ is not } \alpha\text{-rate dom.}) = \sum_{l=0}^{k-1} \sum_{A \in \mathcal{F}_l} \left( \prod_{i \in A} \hat{x}_i \right) \left( \prod_{j \in A^C} (1 - \hat{x}_j) \right). \quad (2)$$

**Theorem 1.**

$$Pr(v_i \text{ is not } \alpha\text{-rate dom.}) < \frac{1}{2}. \quad (3)$$

*Proof.* Let the random variable  $X$  be the number of neighbours that vertex  $v$  has in  $D$ . Then  $X$  follows Poisson's binomial distribution with parameters  $\widehat{x}_1, \dots, \widehat{x}_{\bar{d}_v}$ :

$$Pr(X = l) = \sum_{A \in \mathcal{F}_l} \left( \prod_{i \in A} \widehat{x}_i \right) \left( \prod_{j \in A^c} (1 - \widehat{x}_j) \right).$$

Showing our goal (3) is equivalent to showing

$$\frac{1}{2} \leq \sum_{l=k}^{\bar{d}_v} \sum_{A \in \mathcal{F}_l} \left( \prod_{i \in A} \widehat{x}_i \right) \left( \prod_{j \in A^c} (1 - \widehat{x}_j) \right) = Pr(k \leq X). \quad (4)$$

So we are looking for a minimum of the right hand side of (4) subject to  $\sum_{i=1}^{\bar{d}_v} x_i \geq k$  (this minimum must exist by continuity and compactness). The minimum will be found when  $\sum_{i=1}^{\bar{d}_v} x_i = k$ ; increasing one of the  $x_i$ s without changing the others will clearly only increase the RHS. (Intuitively, increasing the probability of success in one of the trials, while leaving the others unchanged, can only increase the probability of getting at least  $k$  successes.) Keep in mind that  $0 \leq \widehat{x}_i \leq 1$  for all  $\widehat{x}_i$ . So we may assume that

$$\sum_{v_i \in N[v]} \widehat{x}_i = k. \quad (5)$$

Now we can use the result from [16], Theorem 5, that shows that the tail distribution function of Poisson's binomial distribution attains its minimum in the binomial distribution, i.e. when all probabilities are equal. The theorem states that for two integers  $b$ , and  $c$  such that  $0 \leq b \leq np \leq c \leq n$ , the probability  $P(b \leq X \leq c)$  reaches its minimum where all the probabilities  $p_1 = \dots = p_n = p$ , unless  $b = 0$  and  $c = n$ . Here the  $p_i$ s are the probabilities (or parameters) of Poisson's binomial distribution, and  $n$  and  $p$  are the parameters of the related binomial distribution. We apply that theorem taking the two integers  $b$  and  $c$  to be our  $k$  and  $\bar{d}_v$  respectively. We have that  $p$ , the equal probability is  $\frac{k}{\bar{d}_v}$  from (5), whence  $np$  equals our  $k$ . The theorem gives us

$$\sum_{l=k}^{\bar{d}_v} \binom{\bar{d}_v}{l} p^l (1-p)^{\bar{d}_v-l} \leq Pr(k \leq X).$$

Thus, we will be done if we can show that

$$\sum_{l=k}^{\bar{d}_v} \binom{\bar{d}_v}{l} p^l (1-p)^{\bar{d}_v-l} \quad (6)$$

is at least  $\frac{1}{2}$ . Let  $Y$  be a random variable of binomial distribution with  $\bar{d}_v$  trials each of probability  $p$ . Then observe that in fact  $Pr(Y \geq k)$  is equal to (6) above. The median of  $Y$  is bounded by  $\lfloor \bar{d}_v p \rfloor$  and  $\lceil \bar{d}_v p \rceil$  [18], but  $\bar{d}_v p$  is exactly the integer  $k$ , so  $k$  is the unique median of  $Y$ . It follows from the defining property of medians that  $Pr(Y \geq k) \geq \frac{1}{2}$ , and thus  $Pr(Y < k) < \frac{1}{2}$  and the proof is complete.  $\square$

Hence, the probability is lower bounded by  $\frac{1}{2}$ , and the feasibility follows. Let  $A_i$  denote the event that vertex  $v_i$  is  $\alpha$ -rate dominated and let  $B = \cap_{i=1}^n A_i$  be the event that all vertices are dominated. We use the amplification approach (repeating randomised rounding  $t = O(\log_2 \Delta)$  times), where  $\Delta$

is the maximum degree of the graph as found in [6] which results in  $Pr([x_i = 1]) = 1 - (1 - \hat{x}_i)^t$ . We obtain that the expected value of the solution resulting from randomised rounding, given that event  $B$  happens, (i.e. that the solution is feasible) is

$$\begin{aligned}
E \left[ \sum_{i=1}^n w_i x_i | B \right] &= \sum_{i=1}^n w_i Pr([x_i = 1] | B) \\
&= \sum_{i=1}^n w_i \frac{Pr(B | [x_i = 1])}{Pr[B]} Pr(x_i = 1) \\
&\leq \sum_{i=1}^n w_i \frac{1}{\prod_{j \in N[v_i]} Pr(A_j)} (1 - (1 - \hat{x}_i)^t) \\
&\leq \frac{1}{(1 - 2^{-t})^\Delta} \sum_{i=1}^n w_i (1 - (1 - t\hat{x}_i)) \\
&\leq \frac{t}{(1 - 2^{-t})^\Delta} \sum_{i=1}^n w_i \hat{x}_i \\
&\leq O(\log_2 \Delta \cdot OPT).
\end{aligned}$$

Hence, there exists a particular solution that is within  $O(\log_2 \Delta)$  ratio to the optimal solution. Note that  $C = \frac{1}{(1 - \frac{1}{\Delta})^\Delta}$  decreases monotonically down to  $e$  with increasing  $\Delta$  and assuming that  $\Delta \geq 2$ , the maximum is achieved for  $\Delta = 2$ ,  $C = 4$ . A simple randomised rounding algorithm AlgRR follows immediately, by first solving LP and then rounding the solutions to zero or one. This process is repeated  $\lceil \log_2 \Delta \rceil$  times. All vertices with ones then create with high probability an  $\alpha$ -rate domination set with the sum of the weights within  $O(\log_2 \Delta)$  factor of the optimal solution. Finally, if any vertex is still not  $\alpha$ -rate dominated, a required number of its neighbours are added to the solution. We implemented the algorithm in Python, using lpsolve55 [2] through its Python interface to solve linear programmes. We have drawn random numbers from the  $[0, 0.5]$  interval because this has worked better in practice then drawing from  $[0, 1]$ . As solving a linear programme is running in  $O(n^3)$ , i.e. it is quite time-consuming for larger networks, we looked at some alternatives.

### 3.2 Algorithm RRWC

As the range and size of a network determine the size of the linear programme that needs to be solved, we investigated the following strategy. Firstly, we split a network into communities, then we solve a linear programme for each of communities, and use randomised rounding inside communities. Finally, we check if all the vertices are  $\alpha$ -rate dominated, and if not, we add the required number of neighbours of vertices that are still not dominated into the final solution. We implemented this algorithm in Python using NetworkX, and its module [1] based on the Louvain method of community detection given in [3]. In a way, we can look at this as a divide and conquer strategy, where we split the original problem into smaller ones, obtain the smaller problems' solutions based on the previously shown technique, and finally take all those solutions and "patch" them globally in order to obtain a feasible solution for the whole network. We will denote this algorithm **AlgRRWC** (RRWC stands for randomised rounding with communities).

**Input:** A graph  $G$ , a real number  $\alpha$ ,  $0 < \alpha \leq 1$

**Output:** An  $\alpha$ -rate (total positive influence) dominating set  $D$  of  $G$

```

1: Initialize  $D = \emptyset$ ; violation= 1
2: Split  $G$  into communities  $C_1, \dots, C_k$ ;
3: for all  $C_i$  do
4:   while no-of-runs  $< \lceil \log_2 \Delta(G) \rceil$  and violation== 1 do
5:     solve LP;  $\hat{x} = \text{lp.result}$ ;
6:     for all  $v_i$  do
7:        $r = \text{random.uniform}(0, 0.5)$ 
8:       if  $r < \hat{x}_i$  then
9:         add  $v_i$  to  $D$ 
10:      end if
11:    end for
12:    violation=0
13:    for all  $x_i$  do
14:      if  $|N[x_i] \cap D| < \lceil \alpha * \bar{d}_v \rceil$  then
15:        violation= 1
16:      end if
17:    end for
18:    no-of-run++
19:  end while
20: end for
21: for all  $v_i$  do
22:   if  $l = \lceil \alpha * \bar{d}_v \rceil - |N[v] \cap D| \geq 0$  then
23:     add first  $l$  neighbours not already in  $D$  to  $D$ 
24:   end if
25: end for
26: return  $D$ ; {  $D$  is a low-weight  $\alpha$ -rate dominating set}

```

Algorithm 2: Algorithm RRWC for finding a low-weight  $\alpha$ -rate dominating set



## 4 Experimental results

In order to test the performance of those three greedy and two randomised algorithms, we generated three types of random graphs and used some real-life networks obtained from Twitter. We ran the experiments on a 64-bit Windows 7 workstation with Intel i5-2400 CPU at 3.10GHz and 8GB of RAM.

### 4.1 Random generated graphs

We generated three different types of random graphs, with 100 graphs of each type. They all had a similar number of vertices and edges and were created using methods from the NetworkX [15] package. Weights were assigned uniformly at random from integers between 1 and 71 (including the boundaries). The choice of numbers of vertices, edges and weights was informed by real Twitter networks described in Section 4.2. The weights for all the graphs listed here were created by picking uniformly a random number from 1 to 71<sup>1</sup>. The average descriptive statistics for these networks are given in Table 1.

#### 4.1.1 Random graphs, ER type

Normally used as a benchmark, our first type, **ER graph** is widely known as Erdős-Rényi model[4]. An ER or random graph is obtained by choosing uniformly at random from a family  $\mathcal{G}(n, m)$  of all possible graphs on  $n$  vertices with  $m$  edges [4]. We used `dense_gnm_random_graph` method from NetworkX with parameters  $n = 5000$  and  $m = 50000$  to create those graphs and denote them with ER.

#### 4.1.2 Preferential attachment - high clustering graphs, PN type

We used another NetworkX method `powerlaw_cluster_graph` to create graphs that result in approximate power-law degree distribution and high average clustering (we used parameters  $n = 5000, m = 50000, 0.8$  for probability of triangles)[17]. These graphs are denoted with *PN*.

#### 4.1.3 Planted $l$ -partition graphs, PLP type

Additionally, we created graphs that consisted of several interlinked modules or communities (in our case 5 communities with equal sizes of 1000). In these graphs (also called planted  $l$ -partition graphs [7]) vertices in the same community or subgraph are interconnected with higher probability, in our case  $p_{in} = 0.02$  (this value provides each community similar to other types of graphs density) and vertices of different communities are connected with much smaller probability, in our case  $p_{out} = 0.0001$ . We used `random_partition_graph` NetworkX method. This results in graphs having recognisable modular or block structure - with a lot of links inside those 5 communities and only few links between different communities.

---

<sup>1</sup>The largest weight in the Twitter graphs was 71; we made the weights for our randomly generated graphs comparable.

Table 1: Average statistics for random generated networks: V denotes number of vertices, E number of edges, CC number of connected components in the undirected graphs,  $\delta$  minimum,  $\Delta$  maximum,  $\delta_{avg}$  average degree and K is a weight.

Graph	V	E	CC	$\delta$	$\Delta$	$\delta_{avg}$	$K_{min}$	$K_{max}$	$K_{avg}$
ER	5000	50000	1	6	38	20	1	71	39
PN	5000	49847	1	9	37	20	1	71	39
PLP	5000	50980	1	6	38	20	1	71	39

## 4.2 Twitter mentions networks

We used four undirected weekly graphs from Twitter in period of December 2011 to January 2012 obtained from Datasift, where there is an edge between user A and user B if A tweeted at least one message containing “@B” during that week, and B reciprocated at least once. Those networks had around 5k vertices and around 3.5k edges on average. For each vertex we retrieved its *Klout* score and used it as the weight. The Klout score measures an individual’s influence based on her/his social media activity<sup>2</sup>. It is a single number that represents the aggregation of multiple pieces of data about individuals’ social media activity, based on a score model which is not publicly available [20]. The descriptive statistics of the Twitter mentions weekly graphs are given in Table 2 below.

Table 2: Twitter mentions network statistics: V denotes number of vertices, E number of edges, CC number of connected components in the undirected graphs,  $\delta$  minimum and  $\Delta$  maximum,  $\delta_{avg}$  average degree and K is a klout number.

Graph	V	E	CC	$\delta$	$\Delta$	$\delta_{avg}$	$K_{min}$	$K_{max}$	$K_{avg}$
twitt1	5775	3716	2174	1	16	1.2869	10	71	33
twitt2	5537	3537	2094	1	19	1.2776	10	71	34
twitt3	5279	3434	1957	1	15	1.3010	10	71	34
twitt4	5597	3599	2093	1	16	1.2860	10	71	33

## 4.3 Comparison

In this section we compare the performances of three initial vertex selection strategies for our greedy algorithm: S1, vertices sorted in weights ascending order (AlgG\_W); S2, vertices sorted in ascending order according to the ratio of vertex weight to vertex degree (AlgG\_W/D); S3, vertices sorted in ascending order according to the ratio of vertex weight to the sum of weights of vertex’s open neighbourhood (AlgG.W/Wn).

Tables 3 and 5 contain the average results on all the random types of graphs (denoted with ER, PN, PLP) and results on *twitt1-4* obtained for AlgG.W, AlgG.W/D and AlgG.W/Wn correspondingly.

<sup>2</sup>In Twitter, Klout focuses on retweets of a user’s tweets, their username mentions by other users, their list memberships on other users’ curated lists, the number of followers and the number and frequency of replies i.e. how engaged they are.

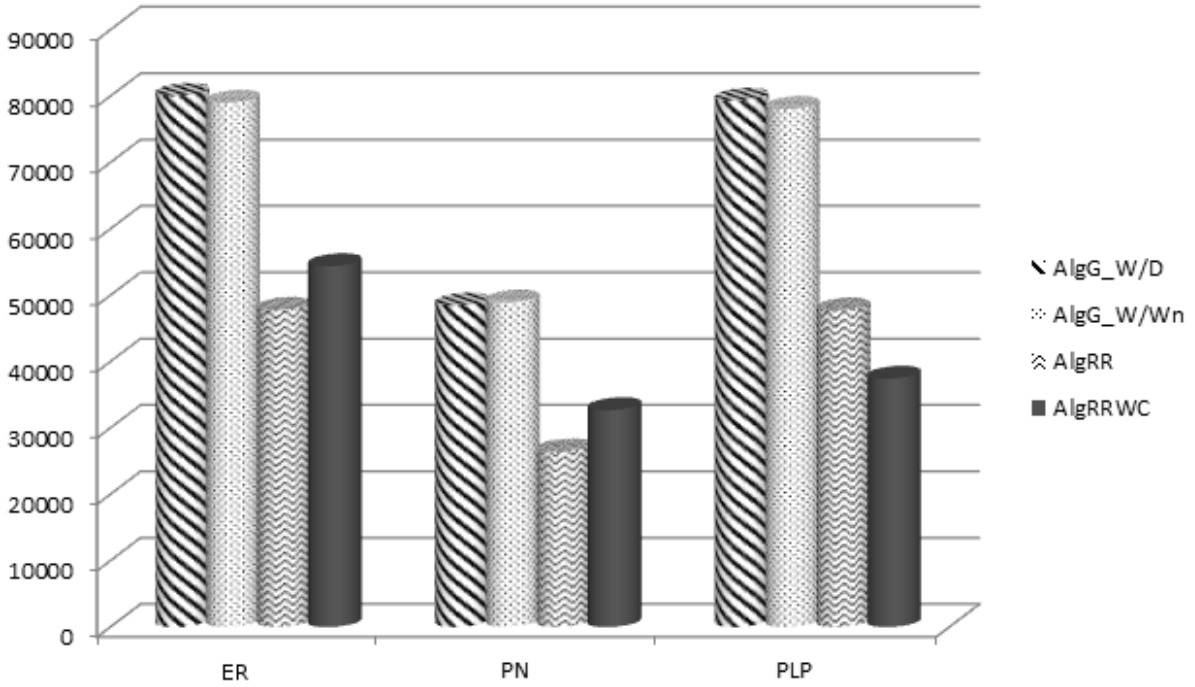


Figure 1: Comparison of sum of weights of 0.25-rate domination sets for AlgG\_W/D and AlgG\_W/Wn, AlgRR and AlgRRWC for random graphs.

Table 3: Alpha-rate domination sets' average sizes (#), average weights (W) and average running times (T) for AlgG\_W, AlgG\_W/D, AlgG\_W/Wn for four different types of graphs (with a set of 100 graphs for each type).

Graph	$\alpha$	AlgG_W			AlgG_W/D			AlgG_W/Wn		
		#	W	Time(s)	#	W	Time(s)	#	W	Time(s)
ER	0.25	2133	82728	8.94	2065	80042	6.04	2037	<b>79007</b>	6.36
ER	0.5	3317	128726	12.67	3205	124507	9.66	3122	<b>121195</b>	9.95
ER	0.75	4343	169062	16.13	4212	163899	13.06	4173	<b>162400</b>	13.31
PN	0.25	1932	74732	8.41	1252	<b>48458</b>	3.86	1263	48918	4.16
PN	0.5	3091	119977	12.03	2430	94319	7.76	2398	<b>93111</b>	7.92
PN	0.75	4160	161757	15.46	3814	148309	11.97	3760	<b>146229</b>	12.13
PLP	0.25	2130	82688	8.75	2043	79322	5.48	2010	<b>78079</b>	6.11
PLP	0.5	3318	128972	12.23	3174	123408	8.67	3119	<b>121345</b>	9.49
PLP	0.75	4327	168590	15.47	4196	163461	11.61	4162	<b>162085</b>	12.55

From Table 3 we can see that S3 strategy wins in all bar one cases (S2 is somewhat better for PN graphs when threshold is 0.25). Also, we can see that while S2 and S3 are comparable, S1 is visibly inferior to both - the average running time is larger while it gives worst results. We can see from

Table 4 that as expected, in random ER and preferential PN type of graphs, especially with the lowest threshold, AlgRR performs better than the alternative version AlgRRWC. However, when the graphs have clear modular structure such as PLP graphs, we see that AlgRRWC outperforms AlgRR and it does it with much smaller average running times. When we compare these results with Table 3 we see that AlgRRWC has about double running times compared with S3 greedy algorithm, but also much smaller weights of  $\alpha$ -rate dominating sets, for lower thresholds and especially for PLP graphs (see Fig 1).

Table 4: Alpha-rate domination sets’ average sizes (#), average weights(W) and average running times(T) for AlgRR and AlgRRWC for random networks. Averages are over 100 graphs. In bold is given the best performing out of two randomised algorithms’ results.

Graph	$\alpha$	AlgRR			AlgRRWC		
		#	W	Time(s)	#	W	Time(s)
ER	0.25	2188	<b>47577</b>	313.49	2214	54373	11.96
ER	0.5	3417	<b>103255</b>	774.76	3379	104245	18.04
ER	0.75	4570	168394	1081.24	4492	<b>165894</b>	23.44
PN	0.25	1334	<b>26732</b>	64.84	1444	32683	7.72
PN	0.5	2573	<b>69321</b>	147.26	2681	75655	14.27
PN	0.75	4134	<b>144827</b>	276.77	4126	146125	21.77
PLP	0.25	2188	477568	237.02	1921	<b>37431</b>	10.34
PLP	0.5	3423	103552	582.64	3116	<b>87823</b>	16.49
PLP	0.75	4600	170058	843.21	4334	<b>154137</b>	22.79

Table 5 list results for Twitter mention networks. Here S2 is more competitive comparing with S3, it wins in half of the cases, but we see from the results that S2 and S3 produce similar results and their average running times are similar. Again S1 is obviously inferior to both S2 and S3. Greedy algorithms outperform both randomised algorithms (See also Table 6.)

#### 4.4 Conclusions

We presented two types of algorithms for solving the minimum weighted alpha-rate domination problem. Our contributions are threefold: firstly, we propose three greedy strategies for this problem; secondly we propose a new randomised algorithm which uses linear programming formulation on the parts of a graph respectively and then recombines it; and finally, we test all proposed algorithms on some sparse real and three different families of dense random graphs. We were able to identify the winning greedy strategy selecting ‘light’ vertices with ‘heavy’ neighbourhoods on dense random graphs. On the sparse Twitter mention graphs in addition to the previously mentioned, another greedy strategy - using ‘light’ vertices with large degrees performed satisfactorily. Therefore, we were able to reject a simple greedy strategy of always putting neighbours with least weights into dominating set. Instead, sorting neighbours by weight over degree or weight over sum of neighbours’ weights works better. When comparing two randomised algorithms, we have seen that solving linear programmes for the parts of graph and patching the solutions obtained to create a feasible solution works much quicker for the graphs of 5k vertices and larger, as expected. Also as

Table 5: Alpha-rate domination sets' sizes (#), weights(W) and running times(T) for AlgG\_W, AlgG\_W/D, AlgG\_W/Wn for mention networks and  $\alpha = 0.25, 0.5, 0.75$  respectively. In bold is given the best performing out of three strategies.

Graph	$\alpha$	AlgG_W			AlgG_W/D			AlgG_W/Wn		
		#	W	Time(s)	#	W	Time(s)	#	W	Time(s)
twitt1	0.25	3132	93192	6.18	2660	<b>82194</b>	2.07	2713	82253	2.35
twitt1	0.5	3201	95964	9.05	2757	<b>85338</b>	2.41	2805	85414	3.01
twitt1	0.75	3491	107624	7.63	3198	100095	2.72	3198	<b>99369</b>	2.82
twitt2	0.25	3001	89439	5.66	2543	<b>79072</b>	1.93	2611	79599	2.19
twitt2	0.5	3058	91792	6.27	2625	<b>81687</b>	2.04	2681	82016	2.22
twitt2	0.75	3353	103764	6.37	3057	95896	2.72	3058	<b>95411</b>	2.68
twitt3	0.25	2858	85647	6.69	2414	75224	2.11	2458	<b>75175</b>	2.05
twitt3	0.5	2925	88314	5.84	2490	77694	1.92	2527	<b>77618</b>	2.28
twitt3	0.75	3204	99756	5.91	2922	92349	2.52	2921	<b>91550</b>	2.49
twitt4	0.25	3045	90254	6.15	2539	<b>78919</b>	2.18	2618	79846	2.82
twitt4	0.5	3112	92963	6.03	2638	<b>81908</b>	2.07	2705	82677	2.23
twitt4	0.75	3390	104289	7.37	3088	96407	3.08	3094	<b>95823</b>	2.87

Table 6: Alpha-rate domination sets' average sizes (#), average weights(W) and average running times(T) for AlgRR and AlgRRWC for Twitter mentions networks. In bold is given the best performing out of two randomised algorithms' results.

Graph	$\alpha$	AlgRR			AlgRRWC		
		#	W	Time(s)	#	W	Time(s)
twitt1	0.25	4655	154355	11.31	4657	<b>154411</b>	6.74
twitt1	0.5	4837	<b>160658</b>	11.85	4838	160666	7.00
twitt1	0.75	5665	<b>191648</b>	12.68	5668	191762	8.19
twitt2	0.25	4469	148980	11.19	4468	<b>148941</b>	6.32
twitt2	0.5	4623	154268	10.90	4623	<b>154267</b>	6.51
twitt2	0.75	5431	<b>184481</b>	11.65	5434	184609	7.55
twitt3	0.25	4220	<b>141094</b>	9.58	4223	141228	5.64
twitt3	0.5	4391	<b>146869</b>	9.94	4397	147115	5.86
twitt3	0.75	5159	<b>175854</b>	10.54	5164	176040	6.88
twitt4	0.25	4471	<b>148289</b>	10.75	4471	<b>148289</b>	6.31
twitt4	0.5	4651	<b>154067</b>	11.08	4653	154111	6.56
twitt4	0.75	5468	<b>184486</b>	12.02	5470	184544	7.69

expected, the obtained results were worse than for solving linear programme on a whole graph (except for the graphs with clear community structure, or for very high thresholds of neighbours that need to be in the dominating set). On the other hand, results obtained by this technique are still superior to the greedy solutions for denser random networks, for lower thresholds. For that reason, the algorithm AlgRRWC is useful as a faster alternative to the AlgRR for larger graphs, especially

if they have modular structure, and as slower but more efficient alternative to greedy techniques for denser graphs and lower thresholds. For higher thresholds and very sparse graphs like the ones we obtained from Twitter, we suggest to use one of the two greedy strategies mentioned above.

Regarding future directions, while we were able to prove the approximation ratio of AlgRR, we would like to do the same for AlgRRWC. In addition it would be interesting to explore in more detail the structure and modularity of graphs that result in better performance of AlgRR or AlgRRWC and the level of sparseness at which the greedy techniques outperform them.

## Acknowledgments.

This work is partially funded by the RCUK Digital Economy programme via EPSRC grant EP/G065802/1 ‘The Horizon Hub’. We would like to thank Datasift for providing us with the Twitter dataset.

## References

- [1] T. Aynaud. Community detection. <http://perso.crans.org/aynaud/communities/>, 2009. Online; accessed 24 August 2015.
- [2] M. Berkelaar, K. Eikland, and N. Peter. lpsolve55. <http://lpsolve.sourceforge.net/5.5/>, 2004.
- [3] V. Blondel, J. Guillaume, R. Lambiotte, and E. Mech. Fast unfolding of communities in large networks. *J. Stat. Mech.*, page P10008, 2008.
- [4] B. Bollobás. *Random Graphs*. Cambridge University Press, second edition, 2001. Cambridge Books Online.
- [5] N. Chen, R. Engelberg, C. T. Nguyen, P. Raghavendra, A. Rudra, and G. Singh. Improved approximation algorithms for the spanning star forest problem. In *Proc. APPROX 2007, LNCS*, pages 44–58, 2007.
- [6] N. Chen, J. Meng, J. Rong, and H. Zhu. Approximation for dominating set problem with measure functions. *Computing and Informatics*, 23:37–49, 2004.
- [7] A. Condon and R. Karp. Algorithms for graph partitioning on the planted partition model. *Random Struct. Algor.*, 18:116140, 2001.
- [8] T. Dinh, Y. Shen, D. Nguyen, and M. Thai. On the approximability of positive influence dominating set in social networks. *Journal of Combinatorial Optimization*, 27(3):487–503, 2014.
- [9] J. Dunbar, D. Hoffman, R. Laskar, and L. Markus.  $\alpha$ -domination. *Discrete Math.*, 211:11–26, 2000.
- [10] K.-T. Förster. Approximating fault-tolerant domination in general graphs. In *ANALCO*, pages 25–32. SIAM, 2013.
- [11] A. Gagarin, A. Poghosyan, and V. Zverovich. Randomized algorithms and upper bounds for multiple domination in graphs and networks. *Discrete Applied Mathematics*, 161(4-5):604 – 611, 2013.

- [12] A. Gagarin, A. Poghosyan, and V. E. Zverovich. Upper bounds for alpha-domination parameters. *Graphs and Combinatorics*, 25(4):513–520, 2009.
- [13] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.
- [14] C. Greaves, P. Reddy, and K. Sheppard. Supporting behaviour change for diabetes prevention. In P. Schwarz, P. Reddy, C. Greaves, J. Dunbar, and S. J., editors, *Diabetes Prevention in Practice.*, pages 19–29. Dresden: Tumaini Institute for Prevention Management, 2010.
- [15] A. Hagberg, D. Schult, and P. Swart. Exploring network structure, dynamics, and function using networkx. In *Proceedings of the 7th Python in Science Conference (SciPy2008), Pasadena, CA USA*, pages 11–15, August 2008.
- [16] W. Hoeffding. On the distribution of the number of successes in independent trials. *The Annals of Mathematical Statistics*, 27(3):713–721, 1956.
- [17] P. Holme and B. J. Kim. Growing scale-free networks with tunable clustering. *Phys. Rev. E*, 65:026107, 2002.
- [18] R. Kaas and J. Buhrman. Mean, median and mode in binomial distributions. *Statistica Neerlandica*, 34(1):13–18, 1980.
- [19] R. Klasing and C. Laforest. Hardness results and approximation algorithms of k-tuple domination in graphs. *Inf. Process. Lett.*, 89(2):75–83, Jan. 2004.
- [20] Klout. "[http://klout.com/corp/klout\\_score](http://klout.com/corp/klout_score)".
- [21] F. Molnàr, Jr., N. Derzsy, E. Czabarka, L. Székely, B. K. Szymanski, and G. Korniss. Dominating scale-free networks using generalized probabilistic methods. *Scientific Reports*, 4(6308), 2014.
- [22] T. Valente. Network interventions. *Science*, 337(6090), 2012.
- [23] D. Vukadinovic Greetham, A. Poghosyan, and N. Charlton. Weighted alpha-rate dominating sets in social networks. In *Tenth International Conference on Signal-Image Technology and Internet-Based Systems (SITIS), Marrakech, Morocco*, pages 369–375, November 23-27 2014.
- [24] F. Wang, H. Du, E. Camacho, K. Xu, W. Lee, Y. Shi, and S. Shan. On positive influence dominating sets in social networks. *Theoretical Computer Science*, 412(3):265 – 269, 2011.
- [25] Y. Wang, D. Chakrabarti, C. Wang, and C. Faloutsos. Epidemic spreading in real networks: an eigenvalue viewpoint. In *Reliable Distributed Systems, 2003. Proceedings. 22nd International Symposium on*, pages 25–34, Oct 2003.
- [26] W. C.-K. Yen. Algorithmic results of independent  $k$ -domination on weighted graphs. *Chiang Mai Journal of Science*, 38:58–70, 2011.
- [27] F. Zou, Y. Wang, X.-H. Xu, X. Li, H. Du, P. Wan, and W. Wu. New approximations for minimum-weighted dominating sets and minimum-weighted connected dominating sets on unit disk graphs. *Theoretical Computer Science*, 412(3):198 – 208, 2011.